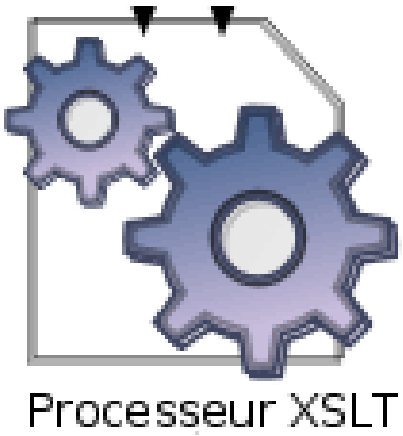
XML and Web Technologies - 201819 [INFO-H509]



Hypothesis report

Project 2:XSLT

Authors:
Siéwé Kouéta Anicet : 00364245

Professor:
Stijn Vansummeren

UNIVERSITÉ LIBRE DE BRUXELLES ULB

April 12, 2019

Contents

1 Introduction

this report aims to give an explanation detailing the choice of our hypothesis for the construction of the **XSLT 2.0** style sheet as well as to explain certain rules and constraints to respect in order to build and generate our HTML files.

2 Structure of the xslt code

For a better approach to the problem, the xsl code written in the **generate-author-pages.xslt** file is divided into 3 parts:

- The creation of Global variables
- The main hand
- Template creation

In each part, we have used a set of xsl function so the most relevant will be explain.

2.1 creation of Global variables

We have created two variables that can be used to handle global. each variable contains an item list.
The variable <**xsl:variable name="SelectAuthors"**> contains the list of all the <author> and <editor> elements of the **dblp-exceprt.xml** file. To obtain this data, we did an iteration that directs the search to all the nodes of the XML document containing the author or editor node.

```
<xsl:for-each select="*/*/author | */*/editor">
```

And the variable <**xsl:variable name="dblps"**> which contains the sub-nodes <**article**>, <**inproceedings**>, <**proceedings**>, <**book**>, <**incollection**>, <**Phdthesis**>, <**mastersthesis**>, <**www**> and their respective sub-nodes.

```
<xsl:for-each select="*/*">
```

2.2 The Main Hand

This part is the main template '/'. Here we made two other functions <**xsl:call-template name="SelectionYear"**> and <**xsl:call-template name="SelectionCo_auteur"**> which are described in Part 3: Creating Template.
It is in this section that we declared the function xsl allowing the creation of HTML files. For this fact, using the xsl functions we defined a succession of rules :

2.2.1 Deleting name duplicates

This function is used to remove all author or editor names that appear more than once in our **$SelectAuthors** variable create higher .

```
<xsl:for-each-group select="\$SelectAuthors/author | \$SelectAuthors/editor" group-by=".">
<xsl:if test="position() ne 1"/>
```

2.2.2 First Name and Last Name Extraction

For each author or editor, This sort is used to retrieve the first Name and Last Name of the latter. for a current author or editor, the **tokenize()** function combine with **last()** retrieves the last word separated by a space.

```
<xsl:value-of select="tokenize(.,' ')[last()]"/>
```

And store the value obtained in a variable **$LastName**, this value corresponds to the Last Name of author or editor.
Similarly, the function **substringbefore(., $LastName)** takes the full name of author or the editor and subtracts from it the Last Name, the result returned by this function will be stored in a variable **$firstName** and will correspond to the first Name of author or the current editor.
To replace non-alphanumeric characters in a string, the **substring-before()** function encapsulates two other **replace()** functions: the first **replace()** function replaces all nonalphanumeric characters with the '=' character, and the second allows to replace all spaces by the character '_'

```
substring-before(replace(replace(., '[^a-zA-Z0-9 -.]', '='), ' ','.'),\$LastName)
```

2.2.3 Get the first letter of Last Name

To retrieve the first letter of the Last Name, we used the **substring($LastName, 1, 1)** function which, for the string $LastName to pass in parameter, will read a character of the string starting with the first character.
To avoid having two identical directory (one with a capital letter and the other lowercase), all the characters returned by this function will be converted to lowercase using the **translate()** function which encapsulates the function **substring()**.

```
<xsl:value-of select="translate(substring(\$LastName, 1, 1), 'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
'abcdefghijklmnopqrstuvwxyz')"/>
```

2.3 Template creation

we created three more templates

2.3.1 Template 'ListAuthors'

for a given author or editor, it allows you to select any other author or editor that appears in the same publication.

2.3.2 Template 'SelectionYear'

it is used to select and group all the years in which author or the current editor appears. The list of current author or editor is taken in the variable **$SelectAuthors** with the parameter

```
<xsl:param name="P_NameAuthor"/>
```

In addition to the years, this Template also selects for the current author:

- all the titles of his publications.
- the List of all the other author or editor that appear in its publication (for this it uses the Template '**ListAuthors**' above)
- as well as links to this publication.

2.3.3 Template 'SelectionCo$_a$*uteur*'

is used to select in alphabetical order the list of co-authors of current author or editor.

3 explication: building HTML documents

in this part we will detail the sets of rules to use in our project to analyze the XML base and turn them into HTML files. We used two technology :

- XPath: used to extract information from the XML document by writing expressions
- and HTML: used to structure our output documents.

3.1 Illegal HTML character: decimal 150

Some error occurred when trying to use a style sheet to generate an HTML 4.x document and trying to generate some unauthorized HTML character codes in the HTML 4.x specification. The error was due to the fact that HTML 4.x does not define any legal characters for ASCII codes 127-159 (included). To avoid XSL transformation errors with a message of the type "Illegal HTML Character: Decimal 150", we define a **character-map** for all the illegal character 'decimal 150' in order to map them to the 'white' character

```
<xsl:character-map name="html-illegal-chars">
    <xsl:output-character character="&#150;" string=" "/>
</xsl:character-map>
```

3.2 Generation of the HTML page

For each author or editor ', an HTML file is created by doing an iteration that selects each author or editor' uniquely.

```
<xsl:for-each-group select="\$SelectAuthors/author | \$SelectAuthors/editor" group-by="."> .
```

All files are directly created in the 'a-tree' directory. Then place in the directory corresponding to the first letter of the name of author or editor. the file name is obtained using 3 variables: **$firstLetter**, **$LastName** and **$firstName** (described above).

```
<xsl:result-document method="html" href="a-tree/{\$firstLetter}/{\$LastName}.{\$firstName}.html">
```

3.3 The <title> tag

Each html page has the name of author or editor 'current as title of the tab and title of the page. In the html <**title**> and <**h1**> database, we insert this name using the value of the **$NameAuthor** variable (detracts above).
For the title of the tab

```
<title>
<xsl:value-of select="\$NameAuthor"/>
</title>
```

For the title of the page

```
<h1>
 <xsl:value-of select="\$NameAuthor"/>
</h1>
```

3.4 Sort publications

publications should be grouped by year and sort in descending order in a table. To group the publications by year, we use the **groupby** function and to make the decreasing sort we associate it with the xsl **sort** function followed by the **order="descending"** attribute.

```
<xsl:for-each-group select="\$GroupeYears/GroupeYear/year" group-by="replace(., '\.$', '')">
    <xsl:sort select="." data-type="number" order="descending"/>
```

Each line after the dates in the publication table corresponds to a publication and contains a number, a link, the list of other author, the title of the publication, as well as other information depending on the type of publication.

- For the index number, the oldest publications have the lowest number. To obtain this index number, we first count the total number of publications in which the author or editor 'appears.

  ```
  <xsl:for-each select="number(count($GroupeYears/GroupeYear/year))">
      <xsl:variable name="Totalpublishers" select="."/>
  ```

  then performed the following operation:
  Index = **total_number_of_posts - Iteration_position_of_loop + 1**

  ```
  <xsl:value-of select="$Totalpublishers - $Numpublisher + 1"/>
  ```

- For the link, we created a condition that tests if the <**ee**> nodes exist in the dblp file for a given publication.

- To display the names of other author, we used the template '**ListAuthors**'

```
<xsl:if test="ee !='' ">
```

3.5 Sort of Co-author

To be able to add the co-author, we did a search to display all the nodes after the root that contain an author or editor 'which is equal to the current author name.

```
<xsl:for-each select="*[author=$P_NameAuthor] | *[editor=$P_NameAuthor] ">
```

then remove all duplicates, testing if in this list the name of an author or editor 'is different from the name of the current otheur.

```
<xsl:if test="not(node()= $P_NameAuthor)">
```